Practical 1: ANN Training and Prediction

Computational Intelligence Assessment: L1 ANN1

Zijin Hong
2020101911
Mathematics and Applied Mathematics
Jinan University - JBJI
hongzijin@stu2020.jnu.edu.cn

1 Introduction

This is a report of practical project[1]: Artificial Neural Network Training and Prediction, the catalogue including:

- Task 1's training result and the corresponding analysis
- Task 2's prediction and computation, and required answers

2 Task1: Manual Training of an ANN Classifier

To address the problem, we constructed a perceptron comprising a fully-connected layer. We employed the *Sigmoid* function to activate the output as necessitated. The model's optimizer is SGD (Stochastic Gradient Descent) with the BCE (Binary Cross Entropy) loss employed as the loss function for the binary classification task. To accommodate the given dataset, the input dimension of the linear layer was set to 3. This was determined by considering "Study Hrs per Week", "Sleep Hrs per Week", and "Quiz Marks" as the three variables.

2.1 Manual Forward Propagation

The forward propagation process involves the calculation of the output of the perceptron using the input features. The steps involved are:

1. Compute the linear part: $z = X \times W + b$, where W represents the weights, b is the bias, and X is the input.
2. Activate the output using the sigmoid function:

$$\textit{Sigmoid}(z) = \frac{1}{1 + e^{-z}}$$

2.2 Manual Backpropagation

Loss Function to measure errors across all training points and then updating these parameters using the SGD optimizer. The steps are broken down mathematically as follows:

1. Compute the Squared Loss gradient with respect to the predictions:

$$\frac{\partial L}{\partial \hat{y}} = -2(y - \hat{y})$$

2. Compute the gradient of the predictions w.r.t. the linear output:

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

[1]The code is available in https://github.com/Rcrossmeister/CI/tree/main/P1

3. Compute the gradient of the loss with respect to the weights:

$$\frac{\partial L}{\partial W} = X^T \left(\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \right)$$

and bias:

$$\frac{\partial L}{\partial b} = \sum \left(\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \right)$$

4. Update the weights and bias using SGD:

$$W = W - \alpha \times \frac{\partial L}{\partial W}$$

$$b = b - \alpha \times \frac{\partial L}{\partial b}$$

where α is the learning rate.

2.3 Results

Setting the learning rate to 0.0001 and the number of epochs to 3, we trained the model with the provided data and computed the MSE as requested.

Epoch	Loss(1)	Loss(2)	Loss(3)	Loss(4)	Loss(5)
1/5	0.2754	0.3375	0.2881	0.2865	0.3262
2/5	0.2103	0.2770	0.2025	0.2206	0.2466
3/5	0.1895	0.2479	0.1834	0.1991	0.2197
MSE	0.1895	0.2479	0.1834	0.1991	0.2197

Table 1: Five randomly selected training results

To study the efficacy of the assigned SGD training outcomes, we did not establish a fixed seed[2]. As delineated in Table1, we observed varied numerical outcomes. As the result shows: since we use squared loss in our training procedure, the MSE is exactly identical to the loss value. We designate the model from Loss(1) as our predictor model, which can be written as:

$$\text{Result} = w_1 \times \text{Study Hrs per Week} + w_2 \times \text{Sleep Hrs per Week} + w_3 \times \text{Quiz Marks} + \text{Bias}$$

where:

$$w_1 = 0.0099, \quad w_2 = 0.0066, \quad w_3 = 0.0141, \quad \text{Bias} = -0.0071$$

2.4 Further Analysis

The detailed calculation result is shown in Table1. From the results, it's evident that the Squared Loss is decreasing over the epochs. This indicates that the model is learning and adjusting its weights to reduce the error between its predictions and the actual values. However, given the limited data from just 3 epochs, it's difficult to definitively identify the model has reached convergence or not.

2.4.1 Suggestions for Optimization

Data Normalization. Normalize the input data to have zero mean and unit variance. This often helps in training neural networks.

Model Complexity. If the data is not linearly separable, consider using a multi-layer perceptron (MLP) with one or more hidden layers.

Data Augmentation : If data scarcity is a concern, augmenting it might help the model to generalize better.

[2]In PyTorch, a seed is a parameter set to ensure the reproducibility of random number generation for experiments or training runs.

Better Optimizers : Replacing basic SGD with optimizers like Adam or RMSprop, which adjust learning rates adaptively, could lead to faster convergence.

More Epochs. Train the model for more epochs. The loss of the model is continuously decreasing, which means it is probably converging to a optimal result.

3 Task2: Churn Rate Prediction and Analysis

For the churn rate prediction, we constructed an artificial neural network (ANN) with two hidden layers, each containing 6 neurons, which is consistent with the number of features selected in the given data preprocessing step. The activation function used for bewteen hidden layers is the *ReLU* (Rectified Linear Unit) function. The output layer contains a single neuron with a *Sigmoid* activation function, suitable for binary classification tasks.

3.1 Results

The model was trained using the Adam optimizer with a learning rate of 0.001, the loss function used for training is the BCE loss, the number of training epochs in 100. The model was evaluated on a test set[3], for further analysis, we set the threshold value of binary classification to 0.5. The results are as follows:

Metric	Value
Confusion Matrix	$\begin{bmatrix} 1431 & 161 \\ 365 & 40 \end{bmatrix}$
Accuracy	73.70%

Table 2: Evaluation metrics for the ANN model on the test set

3.2 Further Analysis and Discussion

3.2.1 Prediction for a New Customer

Given the details of a specific customer:

Geography	Credit Score	Gender	Age	Tenure	Balance	No. of Products	credit card	Active	Estimated Salary
France	600	Male	40	3 years	$ 60000	2	yes	yes	$ 50000

Figure 1: Detail information of the customer

The model's corresponding output is:

Probability that the customer will leave the bank: **0.4808**

Given the predicted probability of 0.4808, which is less than 0.5[4], the model suggests that the customer is more likely to stay with the bank.

3.2.2 Binary Result

If the predicted probability for the customer is greater than or equal to 0.5, the binary result will be 1 (indicating the customer is likely to leave). The detail analysis will be dicussed later.

[3] Splited into 20% of the full data

[4] The default threshold of binary classification task

3.2.3 Confusion Matrix and Accuracy

Confusion Matrix. The confusion matrix indicates that the model predicted 1431 customers would stay, and they did. It also predicted 365 customers would leave, but they didn't. The model did not quite correctly predict any customers who actually left, indicating potential issues with the model's ability to identify the churn.

Accuracy. The accuracy is 73.70%, which means the model has the probability of 73.70% to correctly identify a new customer about their churn rate.

3.2.4 The Normalizaton

Normalization of data is crucial, especially for neural networks. Normalized data ensures that all input features are on a similar scale, which helps the model converge faster and achieve better performance. Without normalization, features with larger scales can dominate the training process, leading to suboptimal results.

4 Conclusion

This report presented the training and evaluation of two neural network models for different tasks. While the perceptron model for Task 1 showed fluctuating training results, the ANN model for Task 2 achieved a high accuracy but showed signs of bias. We analyze the corresponding result with specific question, and obtain the corresponding conclusions.